Attorney Docket No.
SUNMP360

PATENT APPLICATION

METHOD FOR GRAPHICALLY DISPLAYING HARDWARE PERFORMANCE SIMULATORS

INVENTORS: (1) Sherman H. Yip
140 Orsi Circle
San Francisco, CA 94124
Citizen of USA

(2) Paul Caprioli
575 South Rengstorff Ave Apt. #20
Mountain View, CA 94040
Citizen of USA

ASSIGNEE: SUN MICROSYSTEMS, INC.
4150 Network Circle
Santa Clara, CA 95054

MARTINE & PENILLA, LLP
710 Lakeway Dr., Suite 170
Sunnyvale, California 94085
Telephone (408) 749-6900

METHOD FOR GRAPHICALLY DISPLAYING HARDWARE PERFORMANCE SIMULATORS

By Inventors:

Sherman H. Yip and Paul Caprioli

BACKGROUND OF THE INVENTION

1. Field of the Invention

[1] The present invention relates to hardware performance simulators and, more particularly, to a method for graphically tracking instructions being examined by hardware performance simulators.

2. Description of the Related Art

[2] A performance simulator is a computer program that simulates electronic hardware components to assist an engineer in designing hardware components. The performance simulator can simulate or model hardware components much faster than building and coding actual hardware components. As a result, the performance simulator often serves as a starting point for development of new hardware.

[3] The performance simulator is a powerful tool that allows the engineer to quickly explore the applicability of various hardware designs, to check for design errors, to analyze designs, and to determine performance tradeoffs. In order to check for errors, i.e., debugging, in either the performance simulator itself or the hardware design, the engineer needs to know the internal operation of the individual components in the hardware (and performance simulator) and how they interact. For example, the engineer designs a microprocessor chip architecture that should take three clock cycles to execute a floating point instruction.

However, the performance simulator may erroneously indicate that the proposed architecture is taking four cycles. One way the engineer can check whether the error originated from the performance simulator or from the hardware design is to trace the progression of instructions executing through the simulated hardware components in the performance simulator.

[4] As shown in Figure 1, a prior art approach uses simple diagrams with American Standard Code for Information Interchange (ASCII) text descriptions to show the progression of instructions within the performance simulator. One disadvantage with the prior art diagram is that the textual descriptions are difficult to read. In other words, the engineer must have a thorough knowledge of the hardware design to understand the textual descriptions. Furthermore, the prior art diagram can display only limited textual information. For example, if the textual descriptions for the instructions are long, the prior art diagram must reduce the number of instructions displayed in order to accommodate the long textual descriptions on a display screen. Finally, the prior art diagram is not interactive and the engineer cannot obtain more information than what is displayed on the display screen.

[5] As a result, there is a need for a method for displaying and tracking the progression of instructions within the performance simulator that the engineer can easily understand and providing an interactive display that can accommodate more information.

SUMMARY OF THE INVENTION

[6] Broadly speaking, the present invention fills these needs by providing a method for graphically tracking a progression of instructions through hardware components. It should be appreciated that the present invention can be implemented in numerous ways, including as a process, an apparatus, a system, computer readable media, or a device. Several inventive embodiments of the present invention are described below.

[7] One embodiment provides a method for graphically tracking a progression of instructions through hardware components. Instructions of a code segment are defined by graphical icons where each graphical icon has a displayable appearance that identifies a type of instruction. The method tracks each graphical icon when simulating execution of the code segment through the hardware components. The method then displays a progression of each graphical icon through the hardware components during execution of the code segment.

[8] Another embodiment provides a computer readable medium having program instructions for graphically tracking a progression of instructions through hardware components. The program instructions define graphical icons for instructions of a code segment where each graphical icon has a displayable appearance that identifies a type of instruction. Further, the program instructions track each graphical icon when simulating execution of the code segment through the hardware components. Thereafter, the program instructions display a progression of each graphical icon through the hardware components during execution of the code segment.

[9] Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[10] The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, and like reference numerals designate like structural elements.

[11] Figure 1 is a prior art diagram with ASCII text descriptions that shows a progression of instructions within a performance simulator.

[12] Figure 2 shows an exemplary graphical view of a performance simulator simulating a CPU, in accordance with one embodiment of the present invention.

[13] Figure 3 shows a separate display of information associated with a selection of a graphical icon, in accordance with one embodiment of the present invention.

[14] Figure 4 shows a graphical view of a next progression of instructions at a next clock cycle 30, in accordance with one embodiment of the present invention.

[15] Figure 5 shows a tabular view of a time history of a progression of instructions, in accordance with one embodiment of the present invention.

[16] Figure 6 is a flowchart diagram illustrating the method operations for graphically tracking a progression of instructions through hardware components, in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE EXEMPLARY EMBODIMENTS

[17] An invention is disclosed for a method for graphically tracking a progression of instructions of a code segment through one or more hardware components in a performance simulator. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be understood, however, by one of ordinary skill in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process operations have not been described in detail in order not to unnecessarily obscure the present invention.

[18] The embodiments described herein provide a method for graphically tracking the progression of instructions through hardware components. In order to assist a user to debug the performance simulator or a hardware design, the method uses graphical icons to graphically represent instructions of a code segment. When the code segment is executed in simulation, the method tracks each instruction executed through the hardware components. In turn, the progression of the instructions, represented by the graphical icons, through the hardware components is graphically displayed on a display screen. This graphical display allows the user to quickly visualize the internal operations of the performance simulator.

[19] Embodiments of the present invention can be implemented using C language, C++ language, Java™, or any other programming language. As embodiments of the present invention can be implemented in Java, a brief introduction to Java is provided below. Java is a programming language designed to generate applications that can run on all hardware platforms, small, medium and large, without modification. Developed by Sun Microsystems, Java has been promoted and geared heavily for the Web, both for public Web sites and Intranets. Generally, Java programs can be called from within HTML documents or launched standalone.

[20] Java is an interpreted language. The source code of a Java program is compiled into an intermediate language called "byte code". The byte code is then converted (interpreted) into machine code at runtime. Upon finding a Java applet, the Web browser invokes a Java interpreter (Java Virtual Machine), which translates the byte code into machine code and runs it. Thus, Java programs are not dependent on any specific hardware and will run in any computer with the Java Virtual Machine software. Client applications written in Java can be interpreted, or more commonly, compiled "just in time" to native machine code, and thus achieving the performance of native binaries without sacrificing the portability inherent in Java. On the server side, Java programs can also be compiled into machine language for faster performance. However a compiled Java program loses hardware independence as a result.

[21] Figure 2 shows an exemplary graphical view of a performance simulator simulating a central processing unit (CPU), in accordance with one embodiment of the present invention. In particular, the graphical view shows locations of instructions within hardware components of the CPU at a specific time during execution of a code segment. Generally speaking, executing the code segment of a computer program generates instructions to the CPU. Examples of the instructions include a load instruction, an add instruction, a subtract instruction, a store instruction, a branch instruction, a register movement instruction, a shift instruction, input/output instructions, etc. In one embodiment, graphical icons, like graphical icons 216, 218, 219, and 220, represent the instructions and each graphical icon has a displayable appearance that identifies a type of instruction. As shown, the graphical icons have a displayable appearance of a circular shape with different colors and patterns used for identifying different instructions. However, in another embodiment, the displayable appearance of the graphical icons can be defined by any suitable geometric shape (e.g.,

rectangle, square, circle, triangle, etc.), alphanumeric character (e.g., A,v,t,Q,1,9,10, etc.), symbol (e.g., $,*,@,α,ϯϯ,☼, ♥, etc.), shading, pattern (e.g., solid, hatch, stripes, dots, etc.), and color.

[22] As shown in Figure 2, rectangles graphically represent the hardware components of the CPU. However, any suitable geometric shape can graphically represent the hardware components. The hardware components include an instruction buffer 204, an integer instruction execution pipeline 206, a loads and stores execution pipeline 208, a branch execution pipeline 210, a floating point add execution pipeline 212, and a floating point multiply execution pipeline 214.

[23] The performance simulator can also simulate other suitable hardware components. Exemplary embodiments of hardware components include microprocessors, address switches, data switches, memory controllers, Ethernet, networks, data caches, memory, busses and interconnects, motherboard routing, protocols, etc. The hardware components should be broadly defined to include any suitable hardware attribute of a circuit or a chip. The hardware attribute can include any part that defines the attribute. For example, the hardware attribute can include circuit elements, buses, connections, combination of circuit elements, groups of logic, sub-components, system components, and any related circuit element that glues components together, either physically, logically, or by their communication. As a result, although Figure 2 shows the simulation of the CPU, this is not meant to be limiting, as the performance simulator can simulate the instruction through any hardware attribute of the circuit.

[24] When the code segment is executed, instructions are generated to the CPU and, within the CPU, the execution pipelines 206, 208, 210, 212, and 214 pick corresponding instructions from the instruction buffer 204. For example, the integer instruction execution pipeline 206,

represented by a hatch pattern rectangle, needs to pick add instructions, represented by hatch pattern graphical icons, from the instruction buffer 204. As shown in Figure 2, hatch pattern graphical icons 216 are located in the integer instruction execution pipeline 206. The presence of the hatch pattern graphical icons 216 in the hatch pattern rectangle graphically shows to the user that the integer instruction execution pipeline 206 picked the correct add instructions. In another example, the loads and stores execution pipeline 208, represented by a square pattern rectangle, needs to pick loads and stores instructions represented by square pattern graphical icons. As shown in Figure 2, square pattern graphical icons 218 and 219 are correctly located in the loads and stores execution pipeline 208. However, another hatch pattern graphical icon 220 representing an add instruction is also located in the loads and stores execution pipeline 208. The presence of the hatch graphical icon 220 shows that the loads and stores execution pipeline 208 also picked an incorrect add instruction. Thus, the graphical display assists the user to quickly check visually whether there is an error in the performance simulator or in the CPU design.

[25] Figure 3 shows a separate display of information associated with a selection of the graphical icon 218, in accordance with one embodiment of the present invention. If a user wants detailed information associated with a particular graphical icon, like the graphical icon 218, the user can select the graphical icon 218. The user may select the graphical icon 218 through the use of a mouse, a trackball, a keyboard, a touch sensitive display, or any suitable input device. As shown in Figure 3, the user directs a cursor 326 using a mouse onto the graphical icon 218 and clicks the mouse button to select the graphical icon 218. In another example, the user may touch the graphical icon 218 on a touch sensitive display to select the graphical icon 218.

[26] The selection of the graphical icon 218 causes displays of information associated with the graphical icon 218. In one embodiment, a separate display screen 304 located at the bottom of the graphical display provides textual information associated with the graphical icon 218. In another embodiment, the information displayed is graphical. Exemplary embodiments of information associated with the graphical icon 218 include a name of the instruction 306, an internal representation of the instruction 308, a program counter associated with an instruction 310, a physical memory location of the instruction 312, an instruction disassembly 314, a register source 316, a register destination 318, a virtual addresses 322 of data, and a physical address 324 of the data to be loaded.

[27] The graphical views in Figure 2 and Figure 3 show the performance simulator at a specific instant of time, and specifically, at clock cycle 29, as shown in a small display 222 at the bottom corner of the graphical view. In another embodiment, a progression of each of the graphical icons through the hardware components within the CPU during execution of the code segment can be graphically displayed. In other words, the graphical view can also display movement of the instructions, represented by graphical icons, through the hardware components over a period of time. In an exemplary embodiment, Figure 4 shows a graphical view of a next progression of instructions at a next clock cycle 30. This graphical view shows that the graphical icons are not in the same locations as shown in Figure 2 and Figure 3. For instance, from clock cycle 29 to the next clock cycle 30, the square pattern graphical icon 218 and the hatch graphical icon 220 progressed to different boxes within the loads and stores execution pipeline 208. Essentially, the instructions, represented by graphical icons, are processed in order defined by the code segment. The processing of the instructions enables the display of first processed instructions through later stages of the hardware

components and enables simultaneous display of later processed instructions through earlier hardware stages of the hardware components.

[28] Figure 5 shows a tabular view of a time history of a progression of instructions, in accordance with one embodiment of the present invention. The tabular view displays a hardware location for each graphical icon spanning a period of time. The period of time can be in any time increment (e.g., clock cycles, milliseconds, seconds, minutes, etc.). As shown, textual descriptions symbolize the hardware components. However, in another embodiment, graphical icons can also represent the hardware components. A vertical axis 404 lists the instructions and a horizontal axis 402 lists time increments in clock cycles. In particular, the horizontal axis 402 shows time increasing from left to right. Thus, unlike the graphical view described above which shows graphical icons at a specific instant of time, the tabular view shows the time history of the instructions over a span of time.

[29] For example, a user can locate the location of instruction 57, represented by the graphical icon 218 in Figure 2, at clock cycle 29 by looking up Instr_57 along the vertical axis 404 and looking up clock cycle 29 along the horizontal axis 402 to locate memPipe[1] 406. The hardware component memPipe[1] 406 corresponds to the loads and stores execution branch 208 in Figure 2. As a result, the tabular view provides the hardware location of instruction 57 at a specific clock cycle 29 as well as the hardware locations at clock cycles 25 through 30, in accordance with one embodiment of the present invention.

[30] Figure 6 is a flowchart diagram illustrating the method operations for graphically tracking a progression of instructions through hardware components, in accordance with one embodiment of the present invention. Starting in operation 505, instructions of a code segment are defined by graphical icons where each graphical icon has a displayable appearance that identifies a type of instruction. Thereafter, in operation 510, the method

tracks each graphical icon when simulating execution of the code segment through the hardware components. In order to track each graphical icon, the method essentially monitors the instructions entering and departing from the hardware components, in accordance with one embodiment of the present invention. Finally, in operation 515, the method displays a progression of each graphical icon through the hardware components during execution of the code segment.

[31] With the above embodiments in mind, it should be understood that the invention may employ various computer-implemented operations involving data stored in computer systems. These operations are those requiring physical manipulation of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. Further, the manipulations performed are often referred to in terms, such as producing, identifying, determining, or comparing.

[32] Any of the operations described herein that form part of the invention are useful machine operations. The invention also relates to a device or an apparatus for performing these operations. The apparatus may be specially constructed for the required purposes, or it may be a general purpose computer selectively activated or configured by a computer program stored in the computer. In particular, various general purpose machines may be used with computer programs written in accordance with the teachings herein, or it may be more convenient to construct a more specialized apparatus to perform the required operations.

[33] The invention can also be embodied as computer readable code on a computer readable medium. The computer readable medium is any data storage device that can store data which can be thereafter read by a computer system. The computer readable medium also includes an electromagnetic carrier wave in which the computer code is embodied. Examples

of the computer readable medium include hard drives, network attached storage (NAS), read-only memory, random-access memory, CD-ROMs, CD-Rs, CD-RWs, magnetic tapes, and other optical and non-optical data storage devices. The computer readable medium can also be distributed over a network coupled computer system so that the computer readable code is stored and executed in a distributed fashion.

[34] The above described invention may be practiced with other computer system configurations including hand-held devices, microprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers and the like. Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims. In the claims, elements and/or steps do not imply any particular order of operation, unless explicitly stated in the claims.

What is claimed is: